OOP Introduction

February 11, 2020

1 Introduction to Object-Oriented Programming

1.1 Buzzword Compliant Notes

- responsibility-driven design
- inheritance
- encapsulation
- overriding
- collection classes

1.2 Goals

- knowledege of programming principles
- sound knowledge of OOP
- critical assessement of small sofware systems
- design, build and test well-behaved objects

1.3 Classes

A category or type of thing (compare `struct` in C). A template or a blueprint. Unlike a `struct` in C, can contain both data and *instructions* that may be performed on the data stored in the class. The class should describe all objects of a particular kind in an *abstract* way.

In Java, standard style dicates to start class names with a capital letter.

The *source code* for a class, which must be written following the rules of Java, defines the structure of the class and the behaviour of its methods.

1.4 Objects

An *object* belongs to a particular class and has individual characteristics. An instance of a class.

In Java, objects model objects from the problem domain. For example, for a class grading app, `Student` may be declared as a class. Objects `Alice`, `Bob`, `Charles` represent the individual students in the class. `int Student.average()` could return the grade point average for a particular student.

Unlike a structure in C, a Java class can encapsulate both data fields and methods that act on these.

Object *state* is stored in variables in the class. A class `Student` might have variables like `Student.studentNumber`.

1.4.1 Methods

We communicate with or modify objects by invoking methods on them. An object is usually expected to do something when its methods are called. A method may have a parameter to pass additonal information needed to execute the method.

The method name and the parameter types are called the *signature* of the object. The *header* includes both the type's return type and the signature. A `void` reutrn type indicates that the method returns nothing. Parameters are typed and must specify what type is validly passed to a method.

- `String getName()` is the header of a method in class `Cat()` that takes no parameters. Its signature is `getName()`.
- `void changeName(string NewName)` is the header of a method in class `Cat()` that takes a `String` parameter `NewName`.

Objects can communicate with other objects by calling each other's methods. Objects can also create other objects, and objects can be passed as parameters.

Standard Java style names methods with Camel Case, like `.getBalance()` or `.setQuantumNumber()`.

Mutators (Setters) A method that changes something about a class instance, such as a class variable. In the class `Cat()`, you might use a mutator method `Cat.set_colour()` to indicate that Muffin is a black cat.

Accessors (Getters) A method that returns a class characteristic to the calling procedure - for example, `Cat.get_colour()` which could inform a calling program that Socks is a brown cat.

Constructors A constructor method creates the object and initializes its fields to some set of reasonable values. The constructor is a public method *with the exact same name as the class.*

1.4.2 Public Methods and Variables

Can be manipulated by methods that are not within the class.

1.4.3 Private Methods and Variables

Cannot be accessed from methods outside the class. An implementation of data hiding.

1.5 BlueJ

A bunch of time was wasted trying to learn BlueJ.

- To create a new project: File -> New Project
- Press New Class to get a new class. Right click on the class to edit the class.

1.6 The `Object()` Class

In Java, some methods are common to all possible classes. We can *inherit* these methods from Java's `Object()` class.

1.7 Comment Your Code!!!

[1]:
```
/* This
*  is a
*  multi-line comment.
*/

// This is a single-line comment.
```